



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁷ : G06F 17/30</p>	A2	<p>(11) International Publication Number: WO 00/57311</p> <p>(43) International Publication Date: 28 September 2000 (28.09.00)</p>		
<table style="width: 100%; border: none;"> <tr> <td style="width: 50%; vertical-align: top; padding: 5px;"> <p>(21) International Application Number: PCT/US00/07782</p> <p>(22) International Filing Date: 23 March 2000 (23.03.00)</p> <p>(30) Priority Data: 60/125,923 23 March 1999 (23.03.99) US</p> <p>(63) Related by Continuation (CON) or Continuation-In-Part (CIP) to Earlier Application US 60/125,923 (CIP) Filed on 23 March 1999 (23.03.99)</p> <p>(71) Applicant (for all designated States except US): QUANSOO GROUP, INC. [US/US]; P.O. Box 4116, Wilmington, DE 19807-4116 (US).</p> <p>(72) Inventors; and (75) Inventors/Applicants (for US only): SHAMIM, Asad [-/US]; Fairfax, VA (US). EUBANKS, Sam [-/US]; Arlington, VA (US).</p> <p>(74) Agent: MEHRA, Shailesh; Wilson Sonsini Goodrich & Rosati, 650 Page Mill Road, Palo Alto, CA 94304-1050 (US).</p> </td> <td style="width: 50%; vertical-align: top; padding: 5px;"> <p>(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).</p> <p>Published <i>Without international search report and to be republished upon receipt of that report.</i></p> </td> </tr> </table>			<p>(21) International Application Number: PCT/US00/07782</p> <p>(22) International Filing Date: 23 March 2000 (23.03.00)</p> <p>(30) Priority Data: 60/125,923 23 March 1999 (23.03.99) US</p> <p>(63) Related by Continuation (CON) or Continuation-In-Part (CIP) to Earlier Application US 60/125,923 (CIP) Filed on 23 March 1999 (23.03.99)</p> <p>(71) Applicant (for all designated States except US): QUANSOO GROUP, INC. [US/US]; P.O. Box 4116, Wilmington, DE 19807-4116 (US).</p> <p>(72) Inventors; and (75) Inventors/Applicants (for US only): SHAMIM, Asad [-/US]; Fairfax, VA (US). EUBANKS, Sam [-/US]; Arlington, VA (US).</p> <p>(74) Agent: MEHRA, Shailesh; Wilson Sonsini Goodrich & Rosati, 650 Page Mill Road, Palo Alto, CA 94304-1050 (US).</p>	<p>(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).</p> <p>Published <i>Without international search report and to be republished upon receipt of that report.</i></p>
<p>(21) International Application Number: PCT/US00/07782</p> <p>(22) International Filing Date: 23 March 2000 (23.03.00)</p> <p>(30) Priority Data: 60/125,923 23 March 1999 (23.03.99) US</p> <p>(63) Related by Continuation (CON) or Continuation-In-Part (CIP) to Earlier Application US 60/125,923 (CIP) Filed on 23 March 1999 (23.03.99)</p> <p>(71) Applicant (for all designated States except US): QUANSOO GROUP, INC. [US/US]; P.O. Box 4116, Wilmington, DE 19807-4116 (US).</p> <p>(72) Inventors; and (75) Inventors/Applicants (for US only): SHAMIM, Asad [-/US]; Fairfax, VA (US). EUBANKS, Sam [-/US]; Arlington, VA (US).</p> <p>(74) Agent: MEHRA, Shailesh; Wilson Sonsini Goodrich & Rosati, 650 Page Mill Road, Palo Alto, CA 94304-1050 (US).</p>	<p>(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).</p> <p>Published <i>Without international search report and to be republished upon receipt of that report.</i></p>			
<p>(54) Title: METHOD AND SYSTEM FOR MANIPULATING DATA FROM MULTIPLE SOURCES</p>				
<p>(57) Abstract</p> <p>The invention provides users a simplified view of information stored in, and across, all of their operational systems, which may include data warehouses/marts, relational and non-relational databases and text sources. This enables users to query quickly and easily any cross-enterprise data source for up-to-the-minute information. It allows them access to current information in real-time, regardless of where the data resides. Embodiments of the invention allow users make queries through their web browser and receive immediate, direct, simultaneous access to information from multiple ERP systems, proprietary databases, strategic extranets and the Internet. The invention transposes data formats from different databases into a common format, enabling users to access and use information from different, incompatible databases simultaneously. Scenarios can be saved in the server and integrated into any Internet/Intranet application. As such, the invention integrates business data from incompatible systems, without creating an additional data store. Embodiments of the invention include an object-oriented data retrieval system for retrieving data from a plurality of external data sources. The object-oriented data retrieval system includes a category object encoded in an object-oriented programming language. In embodiments of the invention, the category object may contain a plurality of meta fields and at least one function for ordering data in the plurality of meta fields. The system also contains a plurality of transposed objects encoded in the object-oriented programming language, which contain functions for mapping fields from the plurality of relational data sources to the plurality of meta fields.</p>				
<pre> graph LR CO[Category Object 200] -- contains --> MF[MetaField 202] TS1[(208)] --> TO1[Transposed Object 204] TS2[(210)] --> TO2[Transposed Object 206] TO1 --> MF TO2 --> MF </pre>				

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

METHOD AND SYSTEM FOR MANIPULATING DATA FROM MULTIPLE SOURCES

BACKGROUND OF THE INVENTION

5 **Field of the Invention**

• This invention relates to the field of databases. In particular, the invention relates to an object-oriented system for retrieving data from multiple, disparate sources.

Description of the Related Art

10 In recent months, an ever-growing number of M&A deals have been consummated in "Internet time"...meaning weeks, rather than months or years. The industries in which this condensed deal making time frame has been especially prevalent include technology, telecommunications, pharmaceuticals and financial services.

15 Given these trends, companies wishing to return maximum value to shareholders must find some way of integrating, migrating, exchanging and sharing information across distinct operations, quickly and efficiently. To do this requires a new kind of cross-enterprise aggregation solution that can transcend job functions, helping them effect the "right" deals and avoid the
20 "wrong" deals, all in Internet time.

 Today's semi-manual processes and "point solution" software tools are an inadequate solution to this problem. Companies employing such solutions end up with unusable information, as the information is distributed in a hodge podge of mainframe and distributed databases, and spreadsheets and data files
25 on individuals' computers. This leaves the companies without a cohesive view of the organization, pre or post deal.

 Some companies have recently introduced business-information portals, knowledge portals and web-based enterprise information portals (EIPs). Such portals do provide visibility into a variety of enterprise applications and can
30 access data from multiple databases simultaneously. That said, they are simply not appropriate solutions for the M&A market, for three distinct reasons:

1. these solutions tend to be vertical, not horizontal, in nature, and M&A isn't oriented vertically;
2. these kinds of portals cannot wrest data from several different databases and combine it in one, coherent view— unless significant software coding is involved; and
3. they have no intelligence, sorting or analytical capabilities so they cannot guarantee that they are capturing the latest information, which is critical in M&A.

So, while dozens of EIPs and other applications can access data, they cannot derive the right answers from the data presented and turn those answers into corporate actions. Thus there is a need for a system to integrate disparate data sources expeditiously, with a cohesive view.

SUMMARY OF THE INVENTION

The invention provides users, such as M&A teams and the managers who work for them, a simplified view of information stored in, and across, all of their operational systems, data warehouses/marts, relational and non-relational databases and text sources. This enables them to query quickly and easily any cross-enterprise data source for up-to-the-minute information. It allows them access to current information in real-time, regardless of where the data resides.

Embodiments of the invention allow users make queries through their web browser and receive immediate, direct, simultaneous access to information from multiple ERP systems, proprietary databases, strategic extranets and even the Internet. The invention transposes data formats from different databases into a common format, enabling users to access and use information from different, incompatible databases simultaneously. Scenarios can be saved in the server and easily integrated into any Internet/intranet application. As such, the invention integrates business data from incompatible systems, without creating an additional data store.

Embodiments of the system also include security provisions, an important design point for any M&A software solution. When solutions are shared with other individuals they will have access only to the data and solution

sets that the individuals have permission to use, based on who they are and their role within the organization. The systems administrator may determine, with a few mouse clicks, exactly what information each user can see. The administrator can block access to entire categories of data, such as salary information, or to individual records or fields.

Embodiments of the invention use server-based Java, which enables the system to scale to meet the demands of the largest corporations. Embodiments run on UNIX, Windows NT.

Embodiments of the invention include an object-oriented data retrieval system for retrieving data from a plurality of external data sources. In embodiments, the data sources may be any one or more of the group consisting of a relational database, an object-oriented database, a flat file, a data mart, an ERP application. The object-oriented data retrieval system includes a category object encoded in an object-oriented programming language. In embodiments, the category object is implemented in JAVA. In embodiments of the invention, the category object may contain a plurality of meta fields and at least one function for ordering data in the plurality of meta fields. The system also contains a plurality of transposed objects encoded in the object-oriented programming language, which contain functions for mapping fields from the plurality of relational data sources to the plurality of meta fields.

In an embodiment, the category object contains a function for selecting particular data from the meta fields according to criteria specified by a super user. In embodiments, the data retrieval system is linked to the plurality of relational databases via the Internet. In embodiments of the invention, the data retrieval system also includes an object bank for storing the category object and the plurality of transposed objects. In one such embodiment, the object bank resides on a relational database internal to the data retrieval system.

Embodiments of the invention include a method of retrieving a meta field which includes data from fields in a plurality of databases. The method comprises the steps of (1) selecting the meta field for querying from a category object, wherein the category object contains a plurality of meta fields; (2) accessing a first transposed object which links the category object to a field in a first database corresponding to the meta field; (3) accessing a second transposed

object which links the category object to a field in a second database corresponding to the meta field; (4) looping through the first transposed object and the second transposed object to retrieve information from fields in the first and second databases.

- 5 In some embodiments, prior to looping through the first and second transposed objects, an ordering function in the category object is executed, wherein the ordering function orders data in the meta field according to an ordering metric. In some embodiments the ordering metric is alphabetical. In other embodiments the ordering metric is numerical. In embodiments of the
- 10 invention, the ordering criteria are specified by a super user.

BRIEF DESCRIPTION OF THE FIGURES

Fig. 1 illustrates a layout of the data retrieval system according to an embodiment of the invention.

- 15 Fig. 2 illustrates the use of category and transposed objects according to an embodiment of the invention.

Fig. 3 displays the contents of the Object Database in an embodiment of the invention.

- 20 Fig. 4 depicts a view in the Object Database in an embodiment of the invention.

Fig. 5 depicts an example of an object hierarchy in an embodiment of the invention.

Fig. 6 illustrates the retrieval of information from multiple data sources according to an embodiment of the invention.

- 25 Fig. 7 illustrates an object hierarchy according to an embodiment of the invention.

Fig. 8 depicts an example of an object hierarchy according to an embodiment of the invention.

- 30 Fig. 9 depicts an example of category objects, with corresponding transposed objects and data sources in an embodiment of the invention.

DETAILED DESCRIPTION

A. System Overview

The invention includes a method and system for integrating disparate IT systems. In particular, the invention supports an object-oriented infrastructure
5 that allows data resident on multiple, incompatible data sources, such as mainframe and distributed databases, spreadsheets, and unique data files maintained on individual computers, to be manipulated uniformly and transparently. The invention also enables users to access any information to which they have access rights and assemble the information in a single resource,
10 without the need for special-purpose programming. Additionally, the invention allows such data to be assembled in a single, common format.

An embodiment of the invention comprises a turnkey product, built in the Java programming language. Java code is grouped in classes that direct an application's activities. The system transposes data formats from different
15 databases into a common format in order to access information in different, incompatible databases simultaneously. The system includes a proprietary application server, standard database dynamic data plugs, and application modules, which are described in detail in this application and in U.S. Provisional applications Nos. 60/125,923, filed March 23, 1999; 60/158,685,
20 filed October 12, 1999; and 60/158,850, filed October 12, 1999, which are hereby incorporated by reference in their entirety.

B. Application Modules

The invention includes several applications that collaborate to gather
25 and display business information.

- Director and Migrator applications, which are reserved for the super user. These applications are used to establish access rights and migrate data between data sources
- a Transposer application is also reserved for the super user. This application
30 is used to set up the business tools to retrieve information sought by users.

- a Session Master application, a function reserved for the developer, is used to convert data to HTML applications.
- Organizer, Reviewer, Visualizer, and Profiler are management tools that manipulate and display information to reveal trends and the effects of organizational changes.

The following is a brief description of each application. Not all applications are available to all users. Details of the applications are described in the remainder of this specification.

Director The super user uses Director to establish access rights and privileges for individuals and groups to access information in databases. The super user can deny access to an entire database, restrict access to information belonging to one department, or even block individuals from seeing specific fields in a database record. Director is also used to set up functional groups, the top business category, which are described in further detail in subsequent sections of this specification.

Transposer. Transposer is the core of the system. The super user uses Transposer to build all of the objects that retrieve information from databases and set them up to be used by the business applications.

Organizer. Organizer is a two-dimensional graphical application that helps the super user reassign resources and move information quickly and efficiently. Organizer displays organization charts and allows one to experiment with changes by moving the blocks. Organizer provides navigation capability at the company, department, and employee level.

Reviewer. Reviewer is a reporting and information review tool used to create, view, and share business information.

Profiler. Profiler is used to produce a presentation using graphs, charts, and reports created in the other business applications.

Migrator. Migrator allows transfer of data from one database to another, even when the intended target database is not compatible with the originating database. This function is reserved for the system administrator.

- 5 **Visualizer.** Visualizer provides reporting capability using three-dimensional web-based technology. Visualizer allows you to create, view, and share reports across an enterprise.

- 10 **Session Master.** Session Master adapts the system to operate with HTML and XML web pages. Session Master facilitates mapping data from HTML web pages to incompatible data sources for information retrieval. It automatically scans all HTML code and translates it into instructions in the computer system.

C. Technical Specifications

- 15 Embodiments of the invention may include the following software components. Substitutable equivalents will be apparent to one skilled in the art:

- Oracle 8.0.5 database software (with update patches, if needed, and Web Internet license), or any other relational database or other equivalent database.
- WebLogic Server, version 4.03, with unlimited connections and 128-bit,
20 secure socket layer encryption, or other equivalents apparent to one skilled in the art.
- JDK 1.1.7 (latest version), or another suitable object-oriented programming language.

- 25 Embodiments of the invention may run on the following hardware configurations. Many equivalents will be apparent to one skilled in the art:

- Sun Enterprise 450 server, two (2) 300-megahertz Ultra II Sparc modules. 10 gigabytes of disk space, Eight (8) 500-megabyte RAM memory modules (4 gigabytes total), two (2) redundant power supplies
- Compaq 5500 server or equivalent. two (2) Pentium II 300-MHz process,
30 ten (10) gigabytes of disk space, four (4) gigabytes total RAM, 10/100 TX NIC card (or other appropriate network interface)

D. Application Server

An application server in the invention provides enterprise-level deployment of business information. The key features of the application server include:

- Relative ease of developing and deploying distributed Java applications
- Scaling to permit thousands of cooperative servers to be accessed from tens of thousands of clients.
- 10 • Integrated management environment for comprehensive view of applications resource.
- Transaction semantics to protect the integrity of corporate data.
- Secure communications.
- Rich and flexible application architecture without undue
- 15 complexity.
- Support for component reuse.
- Portability.

In an embodiment of the invention, the application server includes five main components, illustrated in Fig. 1:

- an operating system 100, such as Sun Solaris, Windows NT, or an equivalent,
- a relational database 102 in which the enabling data and Java classes are stored. The database is referred to as an Object Database 102, as it is used
- 25 by the server simply to store objects used in the applications, and is not used by applications or to store data.
- An Information Flow Server 106. The Information Flow Server controls the flow of information among the system's PCs and servers, the source databases, and the Object Database. The Information Flow Server 106 may
- 30 be a WebLogic application server. Other equivalents will be apparent to one skilled in the art.

- An ODBC application 108 that allows the system to manipulate flat files.

E. The Structure of Meta Fields

The present invention includes a data structure referred to as a
5 MetaField. A MetaField is a generic container that holds the meta information
about a field in a database. In an embodiment of the invention, the MetaField is
implemented as a Java object, referred to hereafter as a MetaField Java object.
A MetaField Java Object contains several properties of an object, referred to as
a *category object*. The category object is described in detail in subsequent
10 sections of the present application. When a category object is set up and saved,
the name of the MetaField in a source database is stored in a property named
transObjectField.

Figure 2 illustrates the use of MetaFields in information retrieval. To
draw information from two or more databases 208 210, a separate transposed
15 object is created for each database 204 206 respectively; each of these
transposed objects 204 206 is stored under the category object 200. When
Reviewer or Visualizer sends a query, it loops sequentially through the
transposed objects 204 206, retrieving the information from each database 208
210 (via an SQL statement and the result set), until all transposed objects 204
20 206 are accounted for. The SQL result sets are subsequently merged.

F. Formula/Map

When a field is used to pull or push data, the value of the data that the
field pulls or pushes may depend on certain conditions that depend on the data
25 values of the field itself or on values of other fields. These conditions are
captured by a formula application. The conditions are then imposed by the
formula maps and/or mathematical expressions and attached to the field in
question.

If a field is used to pull data, the formula or map attached to it is termed
30 "in-formula" or "in-map." If the field is being used to push data, the attached

formula or map is termed "out-formula" or "out-map." The application does not allow an in-map and in-formula to be simultaneously attached to the same field. However, all other combinations of in-map, in-formula, out-map, and out-formula are allowed.

5 Any given formula or map attached to a given field consists of three basic elements:

- Incoming data (source data)
- Required conditions
- Outgoing data (display data).

10 With incoming data, a value for the data is read into the record that field may be a part of. The value for this incoming data for a field could be empty, null, or otherwise populated and still have conditions that will generate some outgoing value for the data. The required conditions are those conditions that have to be met in order for the field to generate or store a certain value for the
15 incoming data.

 The outgoing data is data that is generated when all the conditions of the required conditions portion of the formula have been met. Outgoing data can also take the shape of a mathematical expression; for example, some calculation or concatenation that is evaluated and solved in order to achieve a certain value
20 for the outgoing data.

 Achieving this is primarily the function of the following objects:

- *MetaField* This field has the capability to store the FormulaObjectProfile within itself in two places, the source formula and the target formula.
- *FormulaData* This is an object that is used to store a single formula
25 condition. It is populated via the application and is first stored in a vector that is in turn stored in a hashtable contained in the FormulaConditionsObject.
- *FormulaConditionsObject*. This object consists mainly of two hashtables. One hashtable is used to store matrices of formula conditions, while the
30 other stores matrices of expressions (in the form of vectors).
- *MathExpressionObject* This object captures each expression input by the user through the graphical user interface (GUI).

- *FormulaObjectProfile* This contains the formula or map that governs the contents of the Meta Field in which it is placed.
- *ExpressionPFOObject* This object is used exclusively in the evaluation of the math expression. The FormulaObjectProcess determines the way in which this object is used in the expression evaluation process.
- *PrimitiveTypes*. This object converts all currently known primitives in Access, Oracle, and Sybase to Java primitives in an embodiment of the invention. Equivalent databases and object-oriented languages will be apparent to one skilled in the art
- *FormulaObjectProcess* This object evaluates maps, formulas, and expressions. It requires a FormulaObjectProfile (FOP) object to begin processing the value that will be placed in the field that contains the FOP.

The first four objects are used to capture, store, retrieve, and evaluate the formula/map conditions and expressions. The last three are used only to evaluate the formula conditions and expressions.

G. Data Plugs

Data plugs are central to the operation of the information retrieval system. Plugs are middlemen that direct the server to databases. Often these are databases that enterprise resource planners such as PeopleSoft, Oracle HRMS, and SAP are built on. The system plugs connect to any type of relational database or flat file, although some flat files may need conversion to a secure format.

There are two kinds of plugs, read and write. Each read plug is a Java class that represents a data source. A write plug includes two Java classes that convert the data types of retrieved information to the data types of the database that will receive the information, and then deliver the data.

In an embodiment of the invention, read plugs are specific to different databases and flat data files. They are also specific to each installation of the database. For example, a read plug developed for one Oracle database will not work with another Oracle database, even if they are in the same version of Oracle.

Read Plugs

Read plugs retrieve *meta* information from tables in a database. The meta data describes critical features of a table, such as its name, the number of
5 columns, each column's name, the length of data allowed in the column, and the data type (string, integer, floating point, other). In database terminology, column is simply another word for field.

When information is requested from various databases, the read plugs direct the request. An enterprise JavaBean retrieves data through connections
10 established by Information Flow server. Any application server can be thought of as software that reaches out to numerous remote computers— as opposed to the physical computer that the application server resides on.

In the final stage of retrieving information to construct a report or draw a graph, the server executes a structured query language (SQL) statement to
15 access specific tables in one or more databases, process the information, and return a result set in tabular or graphical form. Read plugs supply the targeting information used by the SQL statement.

Write Plugs

Write plugs are used only in Migrator and Session Master. Unlike read
20 plugs, which contain information that locates databases, write plugs are essentially converters. They check the target database to see what type of data is contained under a column heading and change the data type of the incoming data to match the type in the target database. A field set to one data type will not accept a different data type.

25 Data types tell a computer whether an item of information is a number; a string of alphabet letters, perhaps forming a person's name; a calendar date; a time of day; or some other variety of data. The type determines what the computer can do with the information. It can add and subtract numbers, for example, but not the printed characters 1, 3, 100, 233.05 and so on. These
30 would be stored, like letters of the alphabet, as character strings. The data type also reserves a specific amount of memory in the computer to hold the

information. The type "char" holds 16 bits, enough to represent any alphanumeric character or Arabic or Asian ideogram, but not an English Word. That takes a "string" type, big enough to hold several words.

5 A write plug contains two Java classes. The first, Main, starts another Java class on the server that collects information the write plug needs to ship data to a target database. Main gathers the read plugs of the source and the target databases so it can locate the desired tables in the databases. Another parameter tells the write plug the column headings in the target database, where the information will go. A third parameter contains all the data retrieved from
10 the source database.

Once the parameters are gathered, a second class, ProcessPlug, checks the target data types, converts the incoming data types, and prepares and executes the SQL code that places the data in the target database. That code either inserts data into the target database, updates existing data, or deletes data.

15 H. Object Tables

Figure 3 depicts tables maintained on the Object Database 102 that hold objects used by the information retrieval system. The figure contains Table Names 300, their Purpose 302, and corresponding fields 304.

I. Views

20 There are two views in the Object Database 102, which are depicted in Figure 4. The table lists the View Name 400, Purpose 402, and Fields in the view 404. The table also illustrates an example 406.

J. Security

Beyond Java's built-in security features such as secure socket layer
25 (SSL), an embodiment of the invention comes with additional security control. An administrator controls user access to all levels of information, down to the individual field if necessary. For example, a task group may need to see employee profiles but not employee salary. The salary field can be blocked without creating a new view or report.

In familiar fashion, one may create user profiles that contain within them all access rights simply by checking boxes. The user profile carries within it all the necessary access rights and privileges.

5 K. The Information Retrieval Procedure

The information retrieval system retrieves information in the same way in all applications. The user selects an object of interest, for example, a company department in the Organizer application. When the user clicks OK, the system reaches for the information, brings it back, and places it in order 500,
10 as shown in Figure 5.

The request for information goes to the server 600 and is converted to the Result EJB 602. The Result EJB 602 initiates data retrieval. The requested object or objects carry unique ID numbers 604 assigned when the objects are created. The Result EJB 602 picks up the IDs 604 and calls the Object EJB
15 606, which reaches into the object bank 608 and finds the requested objects. The server's current object bank is maintained in Oracle, but any full-strength database is acceptable.

After gathering the objects from the Object Bank 608, the Object EJB brings them back in the form of category objects, transposed objects, and view
20 objects—which are further described infra--along with assigned dynamic data plugs and other information, and passes the information back to the Result EJB 602.

The Result EJB 602 calls the ResultManager EJB 610, which inspects the retrieved objects to see the dynamic data plugs and connection pools
25 assigned to them. It then retrieves the data from the databases 614. At this point, the field mapping carried out when the category object was created comes into play. The ResultManager EJB 610 re-maps the database fields, matching them to Meta Fields. It passes the values back to the Result EJB 602. In the next step, the data is processed with any conversions or formulas, such as a total
30 or an average.

In the final step before returning information to the user's screen, the data goes through a security check where the user's permissions are checked

against the information (down to the field level, if specific fields have been blocked). Approved data is then sent to the user's PC.

Embodiments of the invention are multithreaded. If more than one query is launched, objects are retrieved in parallel.

5 L. Object Hierarchy

In the present invention, objects are arranged in a hierarchy, depicted in Figure 7. An example of an implementation of the objects in the hierarchy is depicted in Figure 8. The object hierarchy includes the following nodes:

- 10 • Functional group 700 is at the top of the hierarchy and is the most general object. An example of a functional group name is a department in an organization, such as Human Resources 800.
- 15 • Functional objects 702 classify information—or objects—that directly relate to the functional group 700. Using the Human Resources 800 functional group example, functional objects could include Applicant Processing 802 and Benefits 804.
- 20 • Business objects 704 classify objects that relate to the functional object 702, for example Applicant 806 and Leave Benefits 808.
- 25 • Category objects 706 classify the information that directly relates to the business objects 704. Again using the Human Resources example, category objects could be Skills 810 and Vacation 812.

Meta Fields are not included in the hierarchy because the fields themselves may be used by any of the category objects. For example, FirstName field may be used in the Skills 810 category object and in the Vacation 812 category object.

- 25 When a category object is used to build scenarios, it brings with it all the characteristics of the category, the data fields, and the instructions contained in the object. For example, the category object *Leave Benefits* 808 contains the instructions that no terminated employee leave data should be retrieved. The result is that when you use *Leave Benefits* to build a Reviewer scenario, the only
- 30 employee leave data brought back from the data base is for current employees. The advantage of using objects is that when the object *Leave Benefits* 808 is

used again in Organizer, one does not have to specify “no terminated employees” again.

Two other parts of the object tree are the fields in the “Category Profile” and transposed objects. As mentioned earlier, fields are not considered objects in the hierarchy because they may be reused in multiple category profiles. Transposed objects are used to link—or map—the data in one database to the data in another database and report on the data in one place, called a view. We’ll use the Applicant category object depicted in Fig. 9. Fig. 9 illustrates metafields in a Category Object 900, alongside the transposed objects 902 which map them to a field in a target database 904. Note that in embodiments of the invention, a metafield in a category object may be mapped to fields in more than one database.

M. Characteristics of Category Objects

The focus of creating objects in the present invention is the category object. Category objects carry most of the information needed to build charts and tables in applications. The category object profile not only contains the field list, but it is also endowed with other characteristics such as type of object, in what applications it is to be used, and whether it is associated with a search object. These characteristics are described in more detail below.

Category Objects Are One Of Three Types

Characteristics are assigned to category objects and fields that are required to make them work in the applications. For example, category objects are used by the Reviewer application to view data in a database. Organizer uses hierarchical objects to depict data from a database in a hierarchical format. Both applications use search objects to search for data in a database. Each of these uses requires a different object type—view object, search object, and hierarchical object.

Category Objects Are Designated For Use In Applications

In embodiments of the invention, the super user determines which objects are used in particular applications. In addition to end-user applications, the super user may allow objects to be used by the Director application to grant
5 permission and access rights to the fields in the objects. To use view objects in Organizer, Reviewer, and Profiler, those names are selected from the application list in the dialog box provided. Search objects may also be used in any of the applications. Hierarchical objects are only used in Organizer to build organization charts, so one would allow access only to Organizer and to
10 Director for hierarchical objects.

Search Objects Are Designated For Use With View or Hierarchical Objects

Search objects contain the instructions required to conduct searches of a database. For example, in Organizer, an organization chart may be constructed
15 based on a supervisor's name that, in turn, was selected from a list retrieved by a search object. In Reviewer, a user may want to report certain information for one or more cost centers. In each case, one creates a search object for the type of search desired and then allows that object to be used as an available search object with a hierarchy or a view. The super user tells the application what set
20 of data is viewed in the application from the available data.

Category Objects May Carry Coded Instructions

Embodiments of the invention may also be used to give instructions to the category object about content that is being pulled from the database. As mentioned earlier, objects can carry code or instructions. For example, one can
25 embed instructions in the object that tell it to only retrieve information about employees who are active or on leave; data about employees who are terminated would not be retrieved. Similarly, one may only want to see the name of an employee's primary supervisor, not the secondary supervisor. The code that the application needs to carry out those instructions is embedded in the category

object. Other instructions for grouping and sorting lists retrieved by search objects can also be coded in the object.

N. Transposed Objects

- 5 Links between the fields in the category object and fields in the database are created and defined in the transposed object. One may create a transposed object for each category object. Each transposed object has a view assigned to it that shows the tables and fields from the database that are to be linked to the fields defined in the category object profile.

O. Conclusion

- 10 The foregoing description of various embodiments of the invention has been presented for purposes of illustration and description. It is not intended to limit the invention to the precise forms disclosed. Many modifications and equivalent arrangements will be apparent.

CLAIMS

What is claimed is:

1. A method of generating objects for organizing and retrieving data from federated data sources, the method comprising:
 - 5 generating a category object, the category object containing a plurality of meta fields and functions for searching and ordering the plurality of meta fields;
 - generating a first transposed object, the first transposed object containing a function which maps a first field from a first external data source to a first meta field in the plurality of meta fields;
 - 10 generating a second transposed object, the second transposed object being of a same object class as the first transposed object, the second transposed object containing a function which maps a first field from a second external data source to the first meta field.
- 15 2. The method of claim 1, wherein the first external data source and the second external data source are relational databases.
3. The method of claim 2, wherein the category object, the first transposed object, and the second transposed object, are implemented in an object-oriented programming language.
- 20 4. The method of claim 3, wherein the object-oriented language is JAVA.
5. The method of claim 4, wherein the category object has access to a second meta field.
6. The method of claim 5, further comprising:
 - 25 generating a third transposed object, which maps a second field from the first data source to the second meta field.

7. The method of claim 3, further comprising:
ordering the first meta field by use of a function in the category object.
8. The method of claim 1, wherein the second external data source is a flat file.
- 5 9. The method of claim 1, wherein the flat file is accessed via an ODBC application.
- 10 10. The method of claim 1, wherein at least one of the first and second external data sources is an object-oriented database.
11. The method of claim 1, wherein the first and second external data
10 sources are any of a data mart, a flat file, a relational database, an object oriented database, an ERP application.
12. A method of retrieving a meta field, the meta field including data from fields in a plurality of databases, the method comprising:
selecting the meta field for querying from a category object, the category
15 object containing a plurality of meta fields, the plurality of meta fields being stored in relational format;
accessing a first transposed object, the first transposed object linking the category object to a field in a first database, the field in the first database corresponding to the meta field;
20 accessing a second transposed object, the second transposed object linking the category object to a field in a second database corresponding to the meta field;
looping through the first transposed object and the second transposed object to retrieve information from fields in the first and second databases.
- 25 13. The method of claim 12, wherein the first and second databases are relational databases.
14. The method of claim 12, wherein at least one of the first and second databases is an object-oriented database.

15. The method of claim 12, wherein the first and second external data sources are any of a data mart, a flat file, a relational database, an object oriented database, an ERP application.
16. The method of claim 12, further comprising:
5 prior to looping through the first and second transposed objects, executing an ordering function in the category object, wherein the ordering function orders data in the meta field according to an ordering metric.
17. The method of claim 16 wherein the ordering metric is alphabetical.
18. The method of claim 16 wherein the ordering metric is numerical.
- 10 19. The method of claim 16 wherein the ordering metric is specified by a user who has access to the category object.
20. A object-oriented data retrieval system for retrieving data from a plurality of relational databases external to the data retrieval system, the data retrieval system comprising:
15 a category object encoded in an object-oriented programming language, the category object containing
a plurality of meta fields
at least one function for ordering data in the plurality of meta fields;
20 a plurality of transposed objects, the plurality of transposed objects encoded in the object-oriented programming language, wherein the transposed objects contain functions which map fields from the plurality of relational databases to the plurality of meta fields.
21. The data retrieval system of claim 20, wherein the category object
25 contains a function for selecting particular data from the meta fields according to criteria specified by a super user.
22. The data retrieval system of claim 20, wherein the data retrieval system is linked to the plurality of relational databases via the Internet.

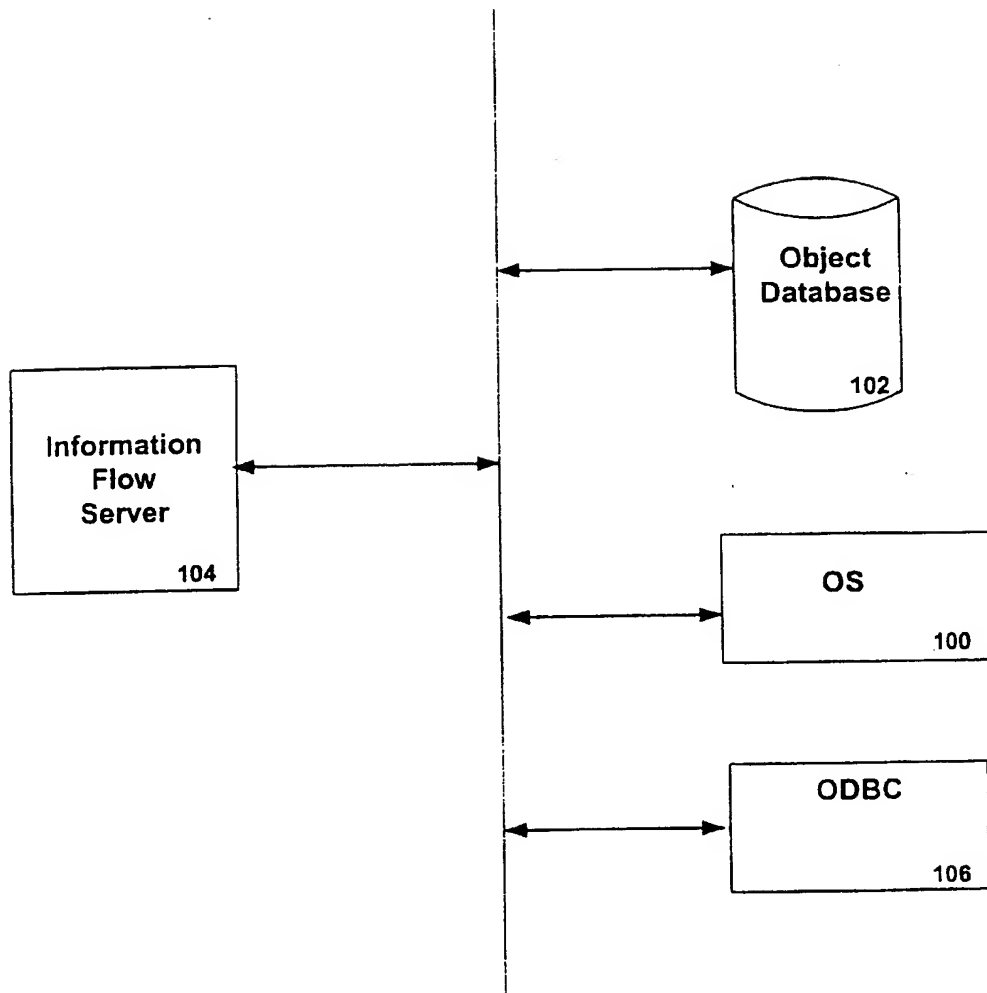


FIGURE 1

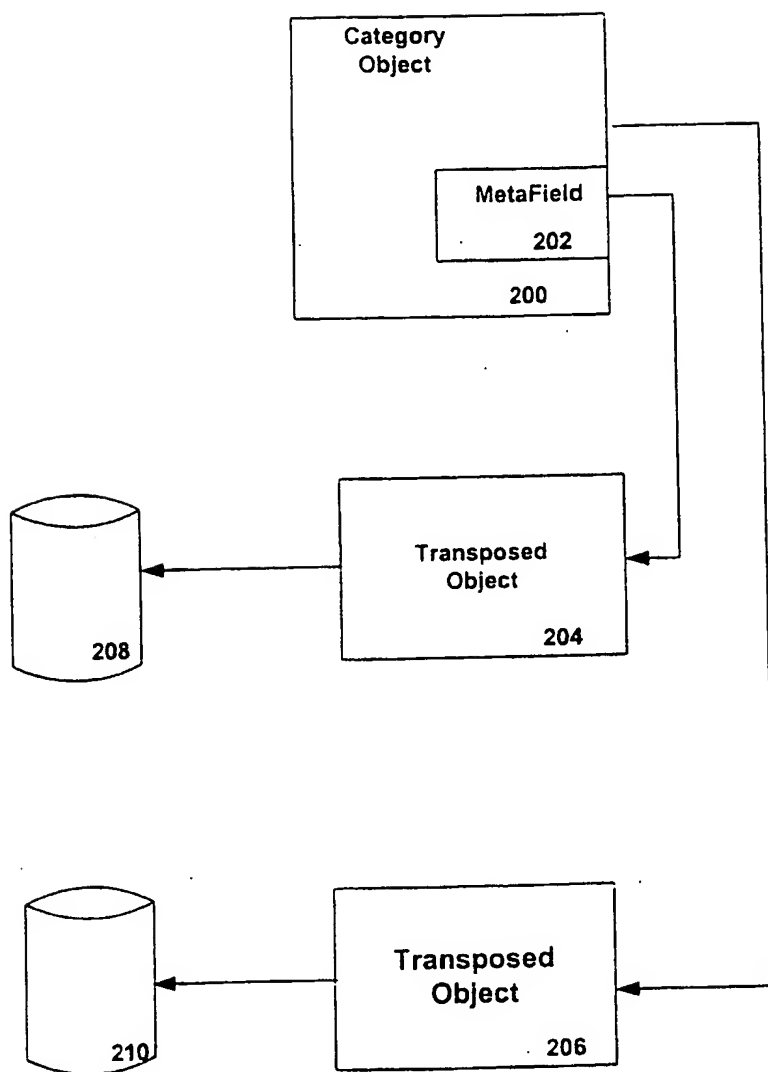


FIGURE 2

300	302	304
Table Name	Purpose	Fields in Table
orchobject	Holds functional objects, business objects, transposed objects, and other such objects.	<i>Field</i>
		ORCHOBJECTID
		ORCHOBJECTBANK
		ORCHOBJECT
orchscednario	Holds reports, Organizer charts, graphs and other scenarios	<i>Field</i>
		ORCHOBJECTID
		ORCHACCESSID
		ORCHOBJECT
orchuserlog		<i>Field</i>
		LOG_ID
		USER_ID
		APP_ID
orch_exceptin_log		<i>Field</i>
		LOG_ID
		USER_ID
		MODULE_TITLE
batch_security		<i>Field</i>
		DESCRIPTION
		DATE_TIME
batch_security		<i>Field</i>
		PLUGID
		USER_ID
		CATID
		CATIDFIELD
		ACCESSID
batch_user		<i>Field</i>
		ALLOW
		<i>Field</i>
batch_user		NAME
		USER_ID
		PASSWORD
		ACCESSTYPE

FIGURE 3

400	402	404	
View Name	Purpose	Fields in View	
colview	Contains the meta data about each column in each table in the customer's databases. Meta data describes the characteristics of each column. A column heading is the same thing as a field.	Field	Purpose
		TNAME	Table name
		COLNO	Column number
		CNAME	Column name
		COLTYPE	Column data type
		WIDTH	Column width, in characters
		NULLS	Can the column contain no value?
recview	List of tables in the customer's databases that Quansoo has access to.	Field	Purpose
		RECNAME	Record available table names

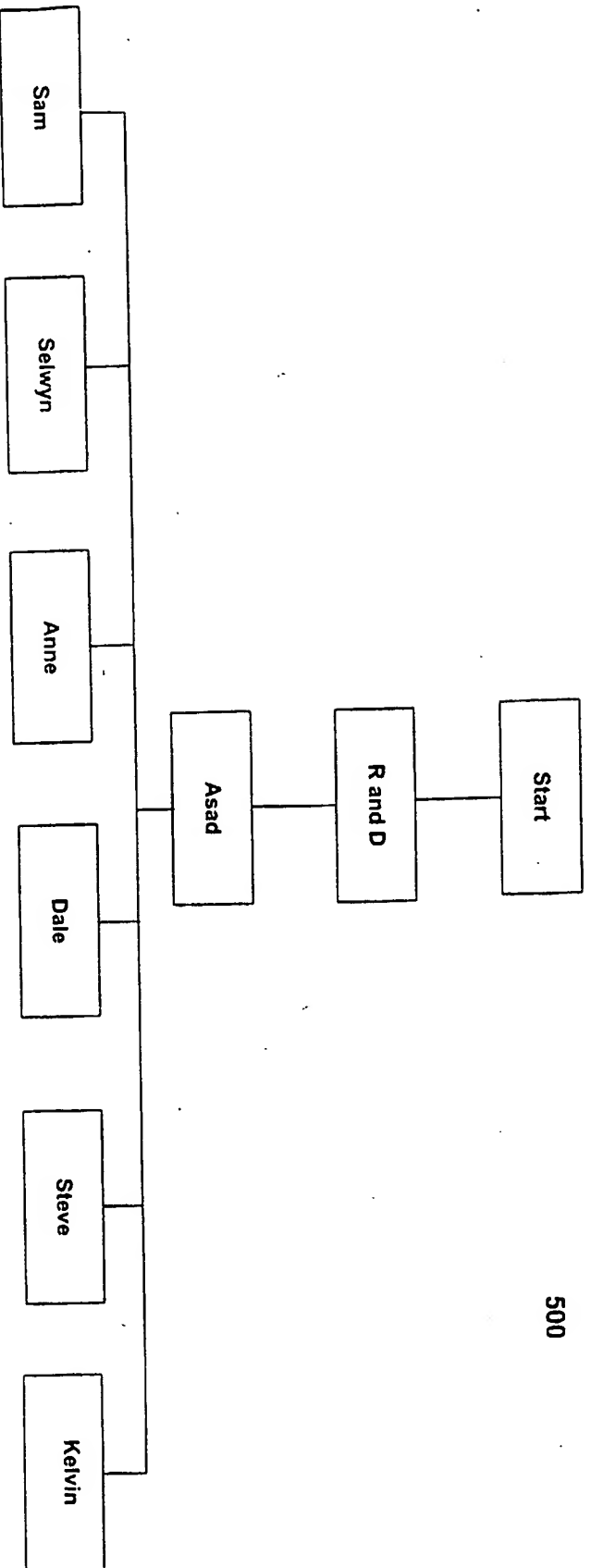
406

TNAME	COLNO	CNAME	COLTYPE	WIDTH	NULLS
Employee_Table	1	ID_NUMBEER	Double	32	No
Employee_Table	2	EMP_NAME	String	15	Yes

FIGURE 4

THIS PAGE BLANK (csp)

Research & Development Dept.



500

FIGURE 5

THIS PAGE BLANK (CONT)

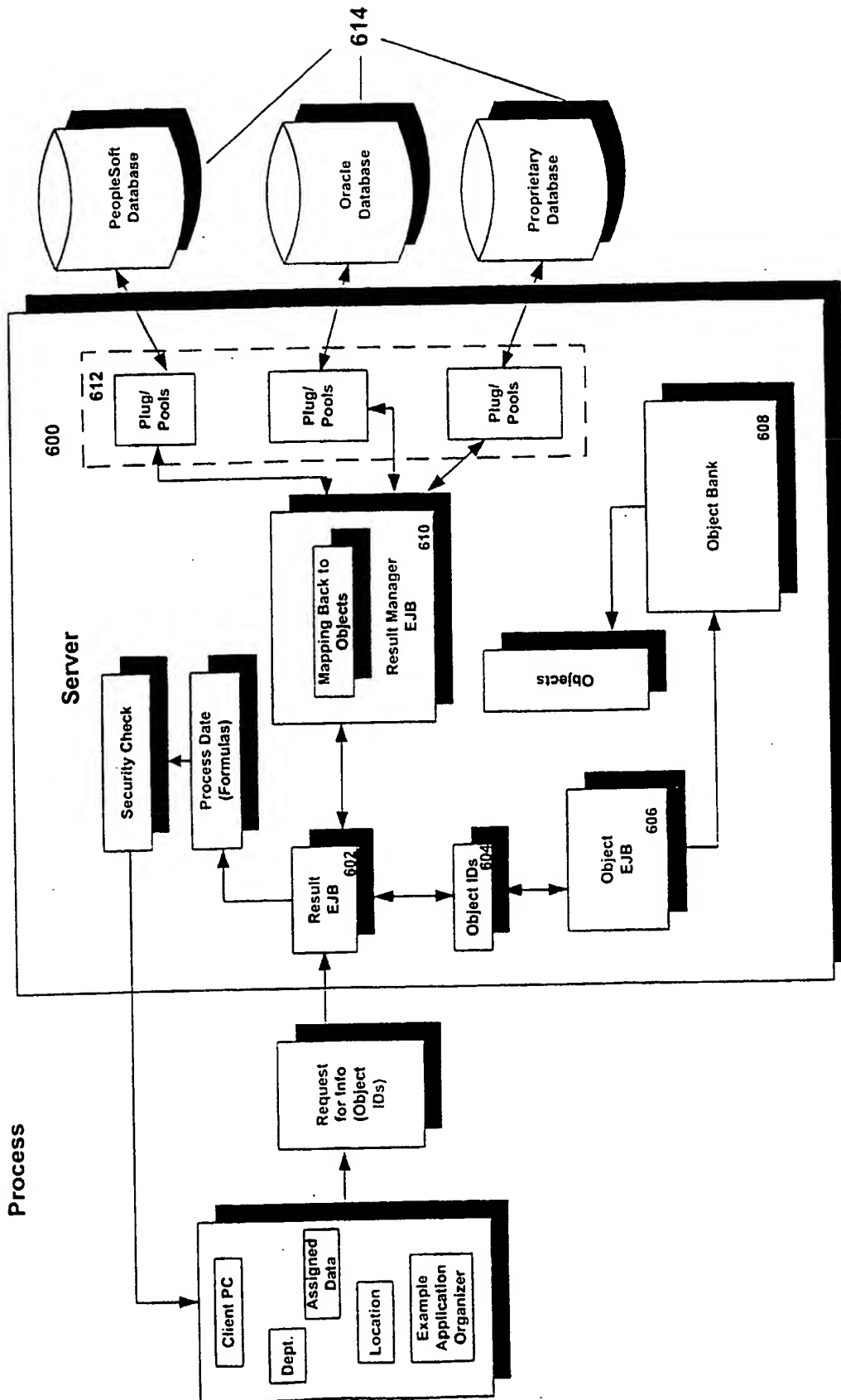


FIGURE 6

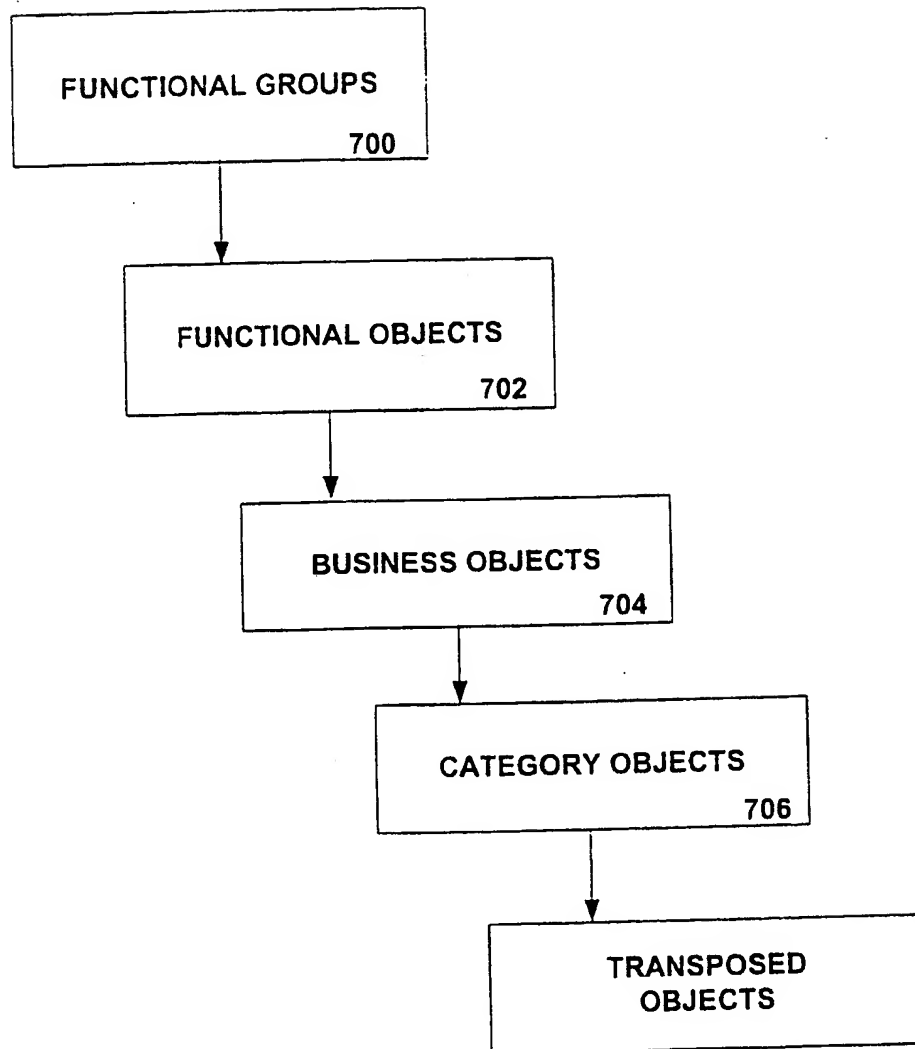


FIGURE 7

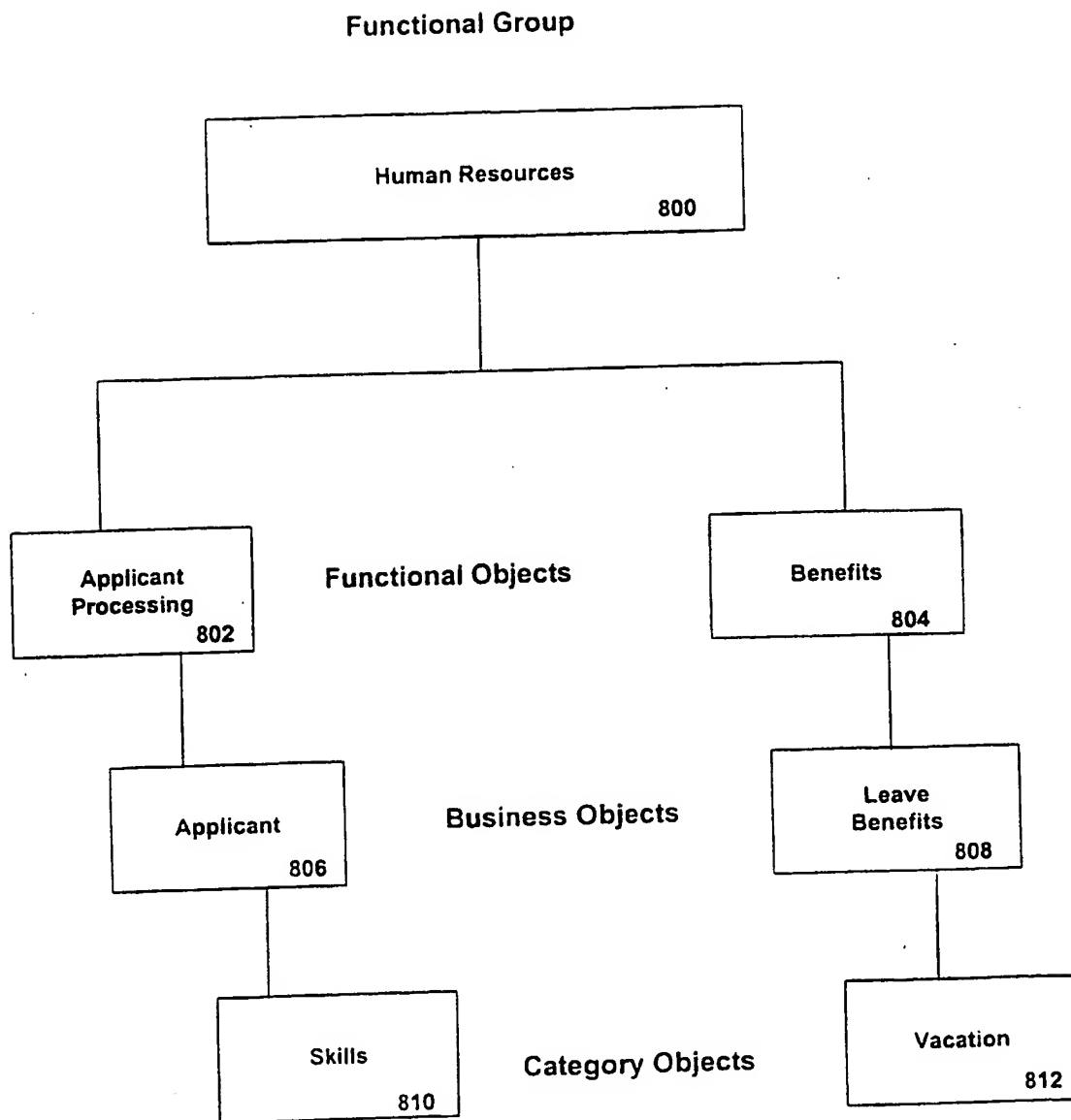


FIGURE 8

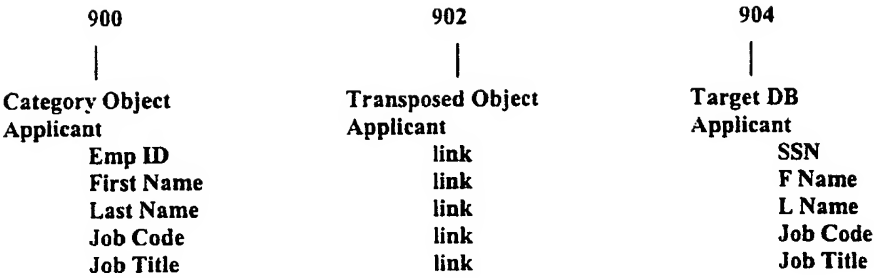


FIGURE 9